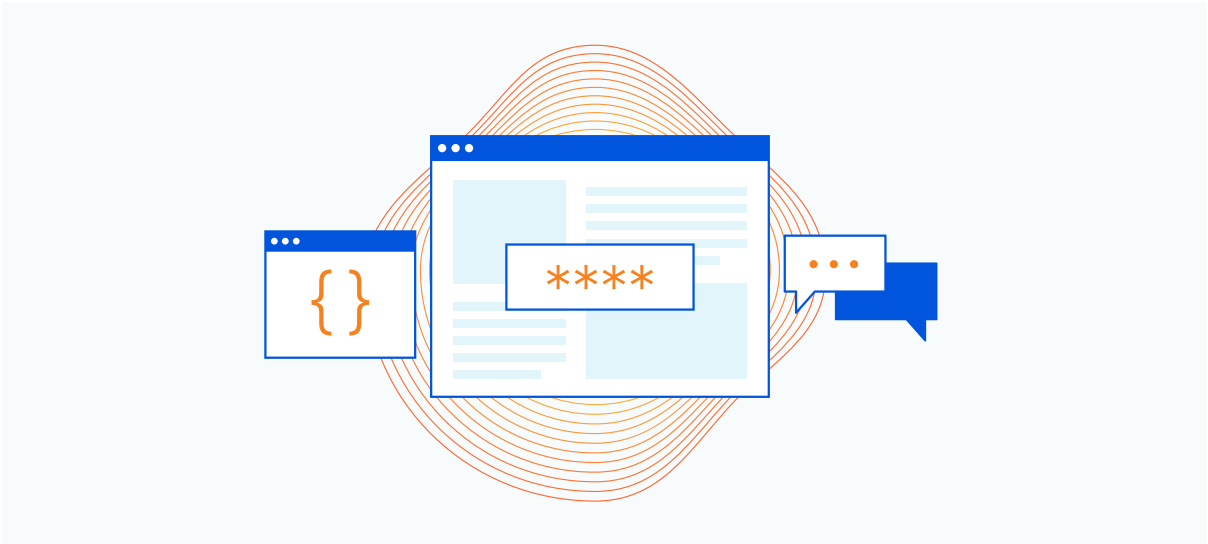
白皮書

CLOUDFLARE

API 安全性指南

+ 886 8 0185 7030 | enterprise@cloudflare.com | www.cloudflare.com

API 使 (App) 世界運轉



我們都知道，在當今社會，是 API（應用程式開發介面）在使世界運轉。更精確地說，它們讓不同的現代應用程式互相通訊。行動或 Web 應用程式可以存取用於儲存和處理資料的後端。API 可以是公用的，允許來自不同公司的應用程式進行通訊，也可以是私人的，用於內部應用程式整合以滿足業務目標。

結果是什麼？更強大、全面的應用程式、網站和行動應用程式，具有更廣泛的功能和更多元化的資料。

例如，共乘公司可以透過付款公司的 API 將其加入，而不需從頭開始建立自己的付款服務。另一個例子是航班彙總網站。為了向我們顯示航班時間、價格、目的地以及我們需要瞭解的所有其他機票相關資訊，他們透過 API 呼叫連線航空公司資料庫，以提取正確的資料，並將其顯示在我們的彙總搜尋結果頁面上。

API 的重要性日益增長，有越來越多公司形容自己為「API 優先」。在某些情況下，公司的實際產品就是一個 API，其商業模式以使用該 API 為中心。例如，如果一家提供天氣資料的公司將他們的 API 作為他們的產品，其他需要天氣資訊的公司將每月支付給他們 API 存取費用。在許多其他情況下，考慮到應用程式預期必須與其他應用程式互動，API 的設計會與提供實際產品功能的程式碼的設計同時進行甚至在其之前完成，而不是在開發結束時加上的。

公司投入時間和精力謹慎設計他們的 API 方法，以考慮如何公開正確的資料，成為收入和商業模式的基礎。

然而，建置完美的 API 並不容易，因為就像任何軟體一樣，可能會出現漏洞而導致我們將在此文件中討論的安全挑戰。

總的來說，API 無所不在，在未來幾年只會加速發展，而且它們必須受到保護。因此，在考慮組織 API 安全性時，我們將深入研究 API 攻擊和縱深防禦的各個方面。

API 漲勢數據

50%

流經 Cloudflare 的流量為 API 流量

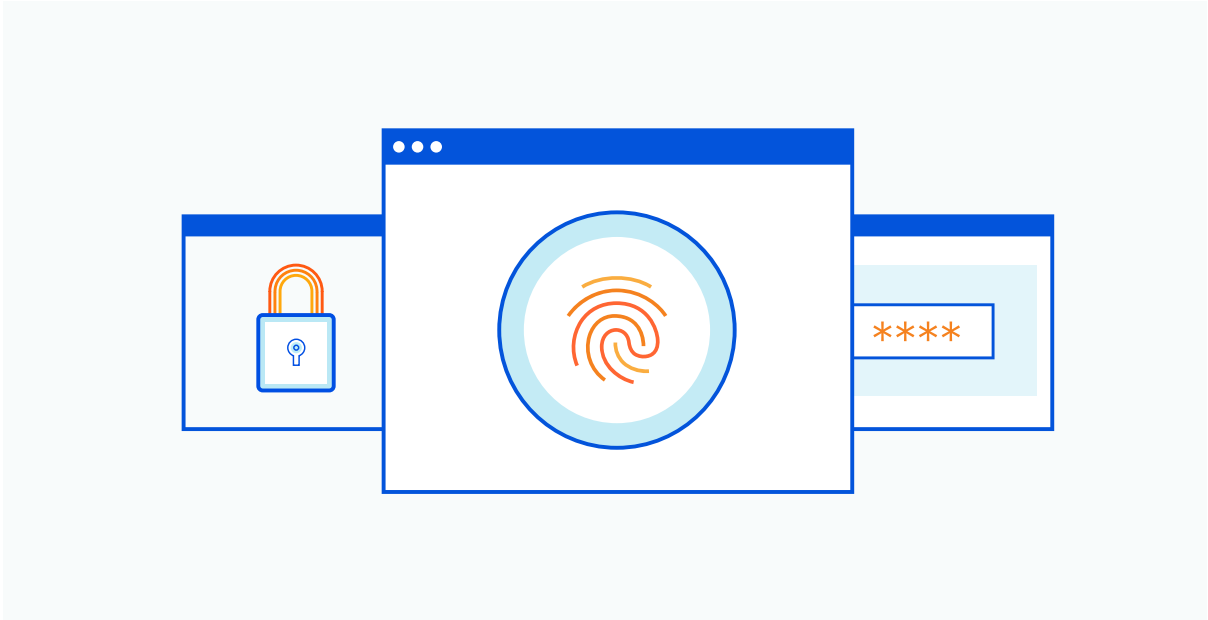
61%

API 流量年度增長

Programmable Web[1] 指出，目前有超過 24,000 個已發布的已知 API。然而事實證明，大多數的 API 都是私人的，用於將內部應用程式連結在一起。私人 API 的數量估計有上百萬。

我們說 API 正在加速發展，而 Cloudflare 就是其成長的直接見證。根據 Cloudflare Radar 2021 年上半年的資料，Cloudflare 網路上大約有一半的流量與 API 有關。更重要的是，它從 2020 年到 2021 年成長了 61%。

考慮到它們公開重要資料，我們需要開始瞭解它們如何帶來我們必須保護的龐大新攻擊面。我們怎麼知道的呢？近年來出現了許多針對 API 的顯著攻擊。



[1]https://www.programmableweb.com/apis/directory

不斷擴大的攻擊面

我們已經知道，API 無處不在，並且是現代商業成功的基礎。它們公開應用程式的邏輯且能夠與其他應用程式分享敏感性資料。

而在所有人意料之中的是，攻擊者也知道這一點，並且蓄意利用企業中這一不斷擴大的攻擊面。

也許 Gartner 是正確的，它們斷言[2]到 2022 年，API 濫用將「從不常見的攻擊媒介轉變為最常見的攻擊媒介，進一步導致企業 Web 應用程式的資料外洩。」

API 是否會成為最受針對的攻擊媒介仍有待觀察，但很明確的一點是，API 將繼續成為攻擊者的目標。

我們說「繼續」，是因為世界已經目睹了一些因 API 安全性不足或 API 開發未考慮安全性而引起的大型外洩事件。

2017 年，T-Mobile 成為 API 攻擊的受害者，有 1,500 萬購買新裝置或申請 T-Mobile 帳戶的客戶資訊遭到洩露。資料包括姓名、地址、出生日期、社會安全號碼、駕照號碼和護照號碼。Vice 報告指出攻擊是透過調整 API 呼叫中的電話號碼參數進行的。任何使用者的電話號碼都可以被查詢，T-Mobile API 會傳送回應，包括被查詢的電話號碼擁有者的敏感帳戶資料。

另一個與 API 相關的外洩攻擊了 USPS，因為攻擊者發現其支援即時包裹追蹤的 API 缺乏基本授權。使用者登入後，可以透過使用萬用字元搜尋參數來查詢任何其他使用者的帳戶資訊，並迫使提供該資料集的所有記錄。這使 6,000 萬 USPS 帳戶持有人的資料處於風險之中。

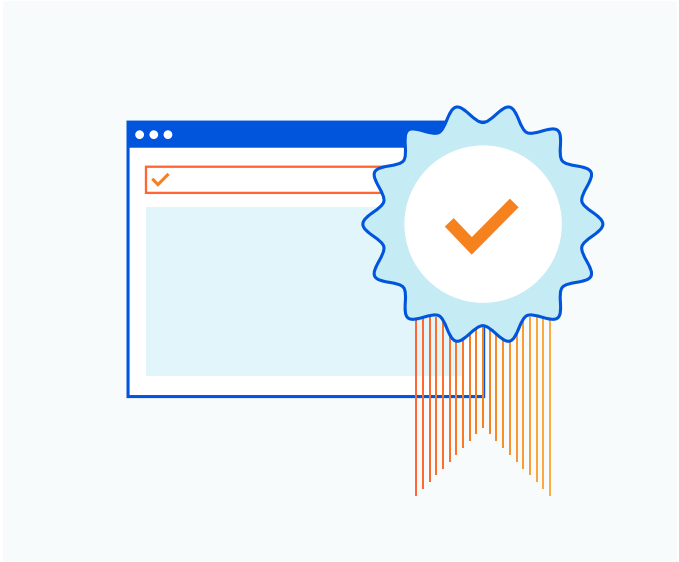
2019 年，印度大型本地搜尋引擎 JustDial，在已被新版本所取代的舊版 API 暴露時，洩露了所有客戶資料。而更糟糕的是，該 API 並沒有設定有效的驗證措施，意味著任何人都可以呼叫 API 並從實際執行伺服器中抓取資料。換句話說，無需複雜手法即可存取使用者資料。

儘管 Facebook 在 GraphQL 方面具有領先地位，但仍因其 API 而多次出現外洩。例如，2019 年底，Facebook 的資料庫被抓取，導致超過 2.6 億使用者的姓名、電話號碼和使用者 ID 陷於風險之中。

在保護 API 方面遇到困難的組織不勝枚舉。原因有幾項。首先，因為不安全的設計而造成的基礎 API 漏洞為攻擊敞開了大門。其次，組織至今還沒有以 API 優先的安全工具。也許他們會使用像是 WAF 或限速之類的 Web 安全工具，但這些工具是為了保護應用程式而不是為了保護 API 而建的。這可能會造成誤判等問題，並明確了為幾乎全自動化的流量設計 API 安全性的需求。

[2]資料來源：Gartner：「API 安全性：您需要做什麼來保護您的 API」，2021 年 3 月

修改版！新的十大 OWASP API



在這其中可以看到的一線曙光是，OWASP 基金會（一個長期專注於改善應用程式安全性的組織）已經參與其中。OWASP 以其「Web 應用程式十大安全風險」而知名，現在他們發布了「API 十大安全風險」，這是一個主要 API 安全性風險和漏洞的清單。

事實上，長期以來我們對應用程式安全性的擔憂，在建置和保護 API 方面也是相同的。

首先，任何以 API 優先的組織在設計 API 時都必須預先考慮安全性。讓我們研究一下剛才提到的一些攻擊，以及它們利用的 OWASP 安全性風險。

十大 OWASP API

1. 無效的物件層級授權
2. 使用者驗證受損
3. 資料暴露過多
4. 缺乏資源與限速
5. 無效功能層級授權
6. 大量指派
7. 安全性設定錯誤
8. 資料隱碼
9. 不當的資產管理
10. 記錄和監控不足

關鍵 API 安全性挑戰

1. 無效的驗證和授權

讓我們從驗證和授權開始，深入檢視上述攻擊利用的幾個關鍵 OWASP API 風險。

JustDial 因其端點上的驗證無效而遭遇攻擊，無效驗證使得任何人都可以發送 API call 質詢。實作驗證後，只有具有正確 TLS certificate, API Keys, Web Tokens 等的 API 呼叫才被允許發出請求，從而顯著降低了 API 安全性風險。

再來看看 OWASP 清單上的第一名，就像我們在 USPS 和 T-Mobile 的例子中所見的，許多 API 攻擊利用效能弱、無效或不存在的授權。無效的物件層級授權很常見——透過將 API 端點有權存取之物件 ID 號碼替換為它們無權存取之物件 ID 號碼，來利用 API 端點。通常只需更改請求中的物件 ID，它們就可以未經授權而存取資料。

API 路徑和查詢參數包括被存取的資源 ID：

經授權的呼叫：

`GET api.greatsampleapis.com/v1/users/235` ，其中 235 是使用者 ID。

被操縱的 API 呼叫可以透過將 235 改成 236 來進行未經授權的存取，即調整物件識別碼（此例子中為 userID）以存取使用者 236 的資料。

`GET api.greatsampleapis.com/v1/users/236`

如果沒有進行授權控制，這可能會成功。開發人員應該對其終端用戶進行威脅行爲建模，以確保攻擊無法調整或修改物件 ID 值以存取其他資料。使用不可預測的物件 ID 值也有幫助，這樣，ID 值就不是連續性的且不易被猜到。

2. 大量指派、資料暴露和資料隱碼攻擊

另一類攻擊在回應中暴露過多資料，或允許使用輸入修改內部物件。

當 API 希望大量公開物件屬性並在回應中傳回過多資料時，就會發生資料暴露，這依賴於用戶端發出請求來篩選資料。

攻擊者可以使用回應中的其他詳細資料來建立更強大的攻擊或網路釣魚電子郵件。例如，如果回應傳回以下所有資料，則其可以撰寫非常具說服力的網路釣魚電子郵件：

```
{
  “Id”: 213,
  “FirstName”: “Sanjay”,
  “LastName”: “Smythe”,
  “EmailAddress”: “ssmythe@hacketyhack.com”,
  “Occupation”: “Assistant to the Deputy Associate Vice Sub-undersecretary”,
  “DOB”: “1986-05-21”,
  “Bank”: “Easygo Financial”,
  “AccountNumber”: 1362886306,
  “PetName”: “Aloysius”,
}
```

當 API 公開內部物件和變數時，大量指派攻擊允許進行 API 呼叫以更改或利用內部值。

OWASP 指出：

> 「軟體架構有時允許開發人員自動將 HTTP 質詢參數繫結到程序碼變量或對象中，讓開發人員更容易使用該架構..... 攻擊者有時可以使用此方法來建立開發人員未曾打算建立的新參數，從而在程式碼中建立或覆寫計劃外的新變量或對象。」

開發人員應該怎麼做？他們應瞭解在開發中呼叫大量指派時的潛在風險，並避免暴露任何可被利用的內部物件或變數。也應考慮可由用戶端更新的 Allowlisting 屬性。

Web 應用程式一直以來都很容易受到資料注入攻擊，API 也不例外。鑑於資料注入攻擊是大家很熟悉的，我們就不在此詳述，但需要強調的是，在傳遞輸入的資料前，應對其進行驗證和清理。組織應採取措施，防止在 API 回應中洩露資料，並限制可回傳的記錄數，以防止大規模暴露事件。

3. 資源濫用和影子/流氓 API

其他攻擊可能會濫用 API，以讓它們消耗過多運算資源——使其不堪重負，而遭受類似 DoS 的攻擊。如果不限制每個用戶端/資源的請求數、單次回應中回傳的記錄數或請求有效承載規模等，就會為這些攻擊敞開大門。

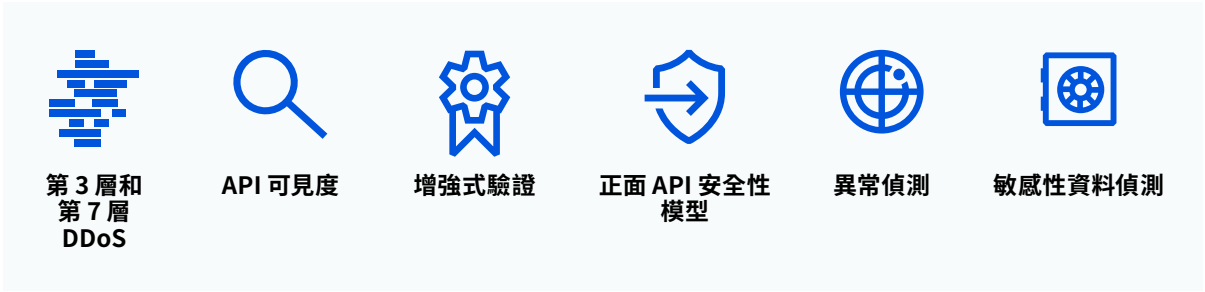
正如我們在 JustDial 攻擊事件中所見，實際執行 API 可能會被遺忘，由於它們很可能未受保護並且可以被利用，因而成為影子或流氓 API。與其他方面的安全性一樣，我們必須瞭解我們的 IT 資產或攻擊面，才能進行適當的網路安全管控。對我們整個 API 端點資產的可見度也是如此。

團隊在開發 API 時，應有完善的流程來追蹤 API 版本，以瞭解實際執行 API 以及已被取代的 API。

保護 API 的注意事項

我們已經介紹了 API 、它們的重要性以及針對 API 的常見攻擊。現在讓我們瞭解一下 Cloudflare 如何建立 API 安全性，以保護 API 免受最常見的攻擊。有效的 API 安全性必須考慮到從可見度、正面安全性模型到阻止濫用等各個方面，從而實現資料保護的。

Cloudflare API Shield



可見度的基礎

可見度

就像網路安全的其他方面一樣，我們首先必須看到要保護的東西，然後才能對其進行保護。API 也一樣，特別是當公司有數百個甚至數千個 API 端點時。

API 探索和可見度是管理 API 的關鍵基礎面，讓組織能夠始終清楚瞭解其 API 端點資產的狀況，以防止影子或流氓 API 成為問題。

就像我們在 JustDial 事件中所見，如果組織無法追蹤 API，就可能會導致資料外洩。

API 縱深防禦

API 第 7 層保護

我們長期以來一直在部署 Web 應用程式防火牆來保護應用程式免受第 7 層 DDoS 攻擊。API 保護應該從許多可靠的控制開始，例如限速和 DDoS 防護，以防止阻斷服務攻擊和蠻力登入嘗試以及由特定 IP 位址執行的一般濫用。這將強制執行 API 使用限制，並對抗 OWASP API 4——缺少資源和限速，確保可用性。

WAF 規則還應該用於識別和封鎖針對 API 的常見攻擊。

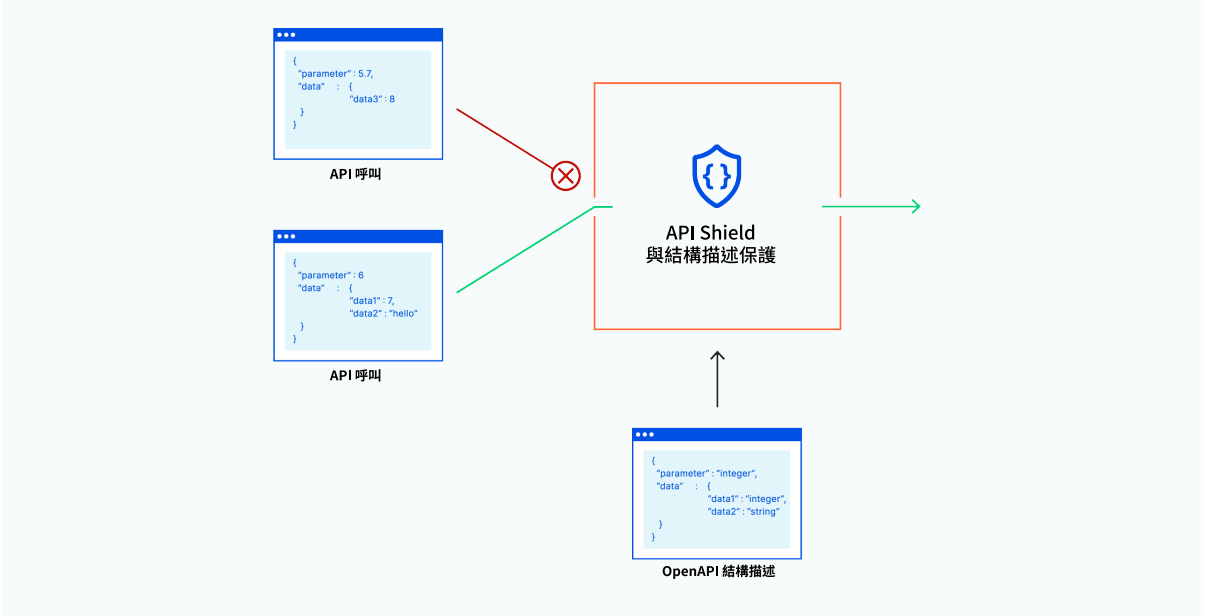
驗證和授權

mTLS 驗證

在我們概述的 API 攻擊中可以看到，缺乏驗證可能是毀滅性的。驗證必須在初始時就內建，並且應使用交互 TLS 加強，以針對行動或物聯網 (IoT) 等用例強制執行基於憑證的身分識別。這種方法是一種更正面的允許清單模型，只允許來自具有有效憑證的合法用戶端的請求連線。

暴露憑證檢查

API 無法避免憑證填充攻擊，這種攻擊會使用被盜的憑證循環嘗試登入。這些帳戶憑證可能會因組織無法控制的第三方外洩而遭到洩露。作為驗證檢查的一部分，API 安全性應該能夠針對外洩憑證資料庫在登入時掃描驗證憑證。如果憑證看起來像是盜用的，API 安全性應該觸發其他安全措施，如重設密碼或多重因素驗證，當然，還有封鎖該嘗試。



結構描述驗證根據 API 結構描述記錄記錄對每個請求進行評估，並封鎖不符合要求的請求。

正面 API 保護

API 結構描述驗證

開發人員竭盡全力地建立 API 結構描述，也就是他們希望其他人如何與 API 互動的文件或基本規則。這可以確立一些事情，例如在每個端點 (GET /users, POST /users) 上的請求方式和操作，或每個操作的輸入和輸出參數。OpenAPI v3，亦稱 Swagger 標準，是用於定義 API 的最知名結構描述。

可靠的 API 安全性應使用在該結構描述上執行的正向 Zero Trust 模型。

有了結構描述後，應根據它自動驗證請求。除了那些已被驗證為符合結構描述的操作外，封鎖其他所有 API 操作。